PATENT CLAIMS

1. A method for the controlled execution of a program (26), the program (26) being intended for a virtual machine (VM, VM'), on a portable data carrier (10), wherein
   - the data carrier (10) has a processor (12) which executes at least a first and a second virtual machine (VM, VM'),
   - the program (26) is executed both by the first and by the second virtual machine (VM, VM'),
   - the operating state of the first virtual machine (VM) and the operating state of the second virtual machine (VM') are checked during execution of the program (26) for correspondence, and
   - execution of the program (26) is aborted if a difference is found between the operating state of the first virtual machine (VM) and the operating state of the second virtual machine (VM').

2. A method according to claim 1, **characterised in that** checking of the operating state of the first virtual machine (VM) and of the operating state of the second virtual machine (VM') for correspondence comprises checking whether the state of a program counter (PC) of the first virtual machine (VM) is the same as the state of a program counter (PC') of the second virtual machine (VM').

3. A method according to claim 1 or claim 2, **characterised in that** checking of the operating state of the first virtual machine (VM) and of the operating state of the second virtual machine (VM') for correspondence comprises checking whether the level of a stack pointer (SP) of the first virtual machine (VM) is the same as the level of a stack pointer (SP') of the second virtual machine (VM').

4. A method according to any one of claims 1 to 3, **characterised in that** checking of the operating state of the first virtual machine (VM) and the operating state of the second virtual machine (VM') for correspondence comprises checking whether the

value of the most recent element (@SP) in a stack (ST) associated with the first virtual machine (VM) is the same as the value of the most recent element (@SP') in a stack (ST') associated with the second virtual machine (VM').

5. A method according to any one of claims 1 to 4, **characterised in that** checking of the operating state of the first virtual machine (VM) and of the operating state of the second virtual machine (VM') for correspondence is in each case performed after an instruction of the program (26) has been executed both by the first and by the second virtual machine (VM, VM').

6. A method according to any one of claims 1 to 5, **characterised in that** the first and the second virtual machine (VM, VM') access a common heap (28) in a non-volatile memory (20) of the data carrier (10).

7. A method according to claim 6, **characterised in that,** when an instruction of the program (26) that contains a write operation to the common heap (28) is being executed, the write operation is performed only by the first virtual machine (VM).

8. A method according to claim 7, **characterised in that** the instruction of the program (26) is executed first by the first virtual machine (VM) and then by the second virtual machine (VM'), and, instead of performing the write operation, the second virtual machine (VM') checks whether the value that is to be written is present in the heap (28) at the location that is to be written to.

9. A method according to any one of claims 1 to 8, **characterised in that** the program (26) is a *Java Card Applet* intended for execution by a JCVM (*Java Card Virtual Machine*).

10. A portable data carrier (10), especially a chip card or a chip module, having a processor (12) and an operating system (22), wherein the operating system (22) has

program instructions for causing the processor (12) to perform a method according to any one of claims 1 to 9.

11. A computer program product having program instructions for causing a processor (12) of a portable data carrier (10) to perform a method having the features of any one of claims 1 to 9.